CAPÍTULO 21
APÊNDICE C: LAYOUTS UNIVERSAIS

No apêndice de Auto Layout estudamos como alinhar e posicionar objetos, de forma que eles mantenham sua posição relativa em diferentes tamanhos de tela.

Ainda dentro do campo de conhecimento de Auto Layout, mas abordando agora um assunto mais avançado, vamos estudar como aproveitar o espaço extra oferecido por alguns dispositivos, como o iPhone 7 Plus em modo *landscape*.

Esse conceito é possível graças ao recurso denominado ***Size Classes***.

21.1 Size Classes

No documento *Human Interface Guidelines* a Apple faz uma importante recomendação de layout de aplicativos, que é estender o uso de elementos visuais para preencher a tela, nos dispositivos que oferecem espaço extra, como o iPhone Plus e os iPads.

Para possibilitar esse recurso, a Apple criou o conceito de *size classes*, que é uma classificação dos dispositivos conforme o seu tamanho, considerando tamanho (*width*) e altura (*height*). Podemos considerar *size classes* como ambientes onde se pode criar conteúdo, conforme o seu tamanho.

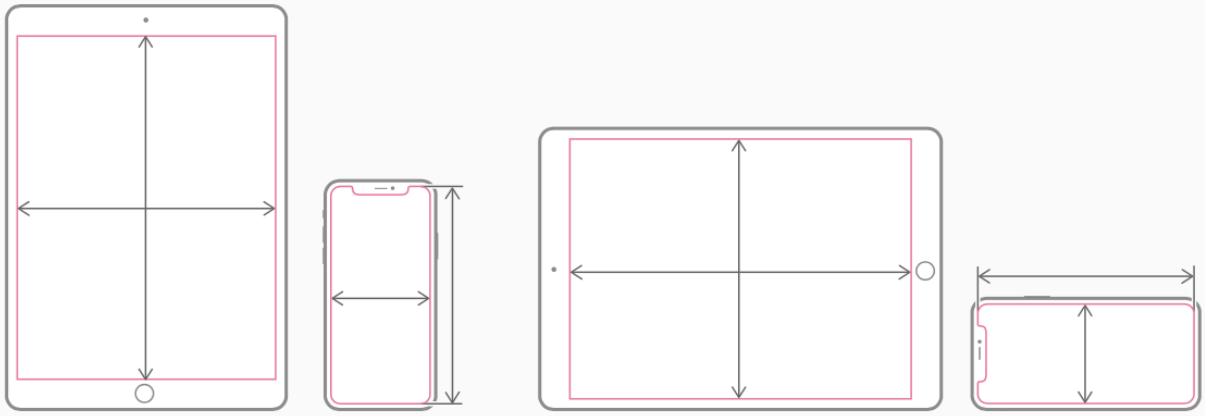
O sistema define dois *size classes*:
- *Regular* - que denota espaço expansível
- *Compact* - que denota espaço restrito

As seguintes combinações podem ser geradas:
- *Regular width, Regular height*
- *Compact width, Compact height*
- *Regular width, Compact height*
- *Compact width, Regular height*

21.2 Os ambientes por dispositivo

A tabela a seguir contém todas as combinações de *size classes*, baseado nos tamanhos de tela o orientação dos diferentes dispositivos.

A fonte desta tabela é o *Human Interface Guidelines*, e já inclui os novíssimos iPhones 8 e iPhone X (a tabela não inclui, porém as telas multitasking dos iPads):



Device	Portrait orientation	Landscape orientation
12.9" iPad Pro	Regular width, Regular height	Regular width, Regular height
10.5" iPad Pro	Regular width, Regular height	Regular width, Regular height
9.7" iPad	Regular width, Regular height	Regular width, Regular height
7.9" iPad mini 4	Regular width, Regular height	Regular width, Regular height
iPhone X	Compact width, Regular height	Compact width, Compact height
iPhone 8 Plus	Compact width, Regular height	Regular width, Compact height
iPhone 8	Compact width, Regular height	Compact width, Compact height
iPhone 7 Plus	Compact width, Regular height	Regular width, Compact height
iPhone 7	Compact width, Regular height	Compact width, Compact height
iPhone 6s Plus	Compact width, Regular height	Regular width, Compact height
iPhone 6s	Compact width, Regular height	Compact width, Compact height
iPhone SE	Compact width, Regular height	Compact width, Compact height

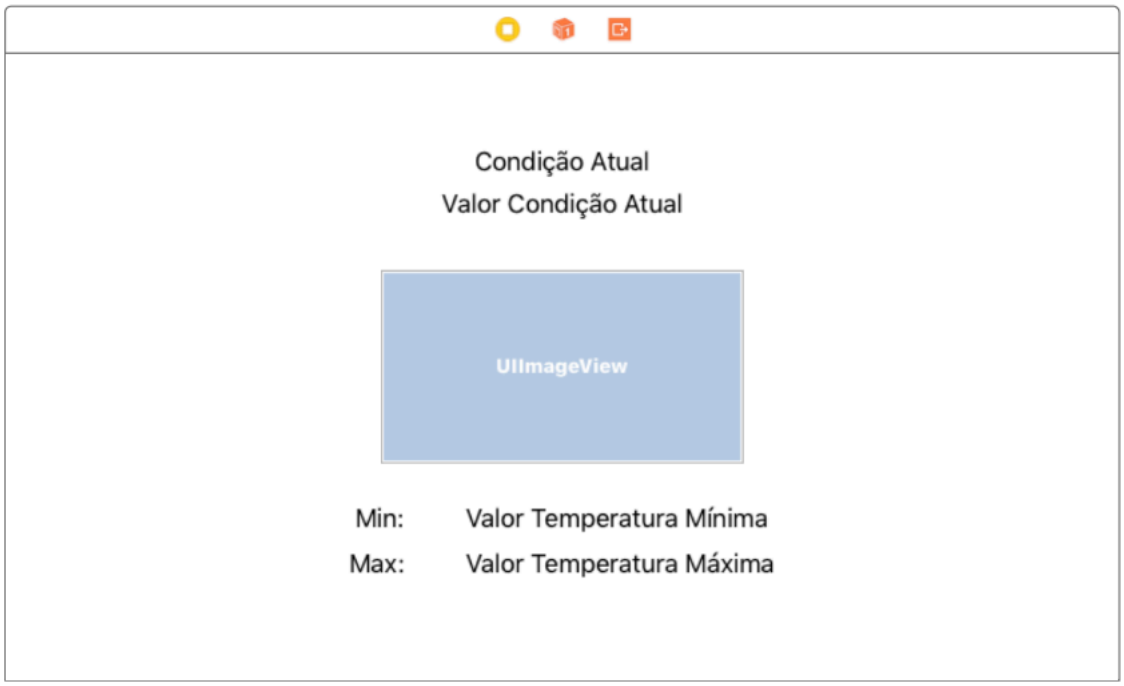
Observando a tabela, podemos concluir que o espaço extra é oferecido nos *sizes Regular*, que nos iPads é oferecido em ambas as orientações, e no iPhones Plus é oferecido em modo *lansdcape*.

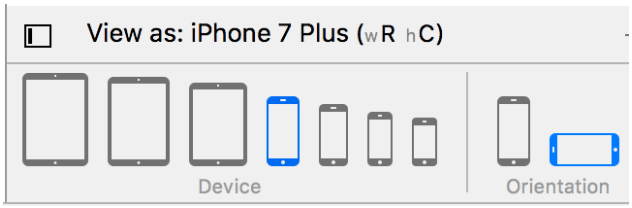
21.3 Onde aproveitar espaço extra em nosso app?

Se observarmos a nossa Storyboard, podemos verificar que a View de Temperatura é a que contém menos elementos visuais.

Se utilizarmos a Live View e variarmos a orientação dos diversos dispositivos, podemos constatar, conforme a figura a seguir, que se colocarmos a *view* em modo *landscape*, no iPhone 7 Plus, teremos um bom montante de espaço extra nas laterais do aparelho.

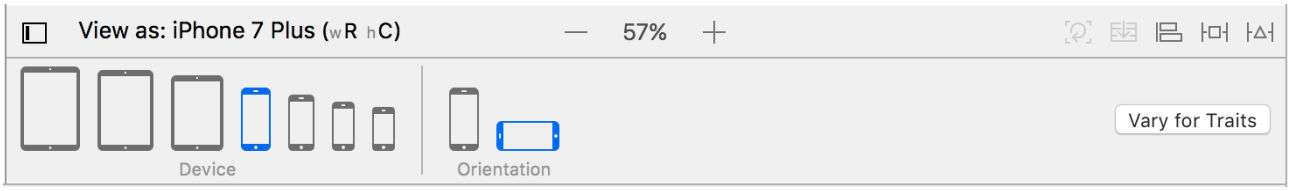
Portanto é neste ambiente que iremos criar novo conteúdo, melhorando assim a experiência daquele usuário que rodar nosso app em um iPhone Plus. Ele terá, portanto, mais informações disponíveis na tela quando colocar o seu dispositivo em modo *landscape*.





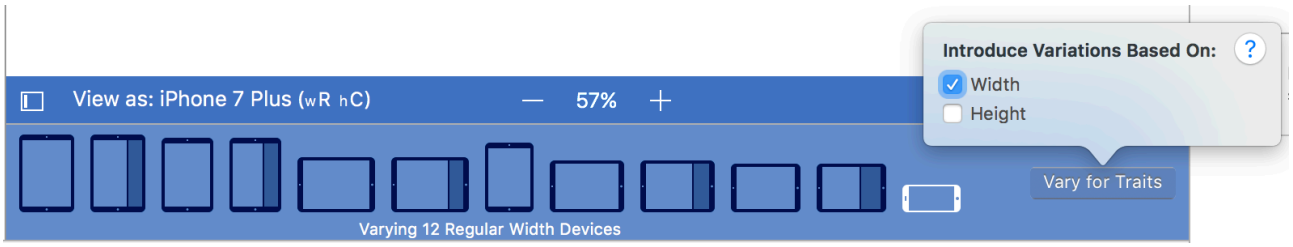
21.4 Como ativar o Size Classes?

O recurso de *size classes* é ativado através da ***Live View***, no Xcode:



Inicialmente você deve escolher em qual dispositivo base para trabalhar, selecionar o botão <Vary for Traits>, e, em seguida, as dimensões desejadas - *width*, *height*, ou ambas. Após essas seleções, o Xcode seleciona todas as variações com os parâmetros selecionadas e coloca a barra na cor azul, indicando que o size classes está ativado.

Veja um exemplo, onde foi selecionada a dimensão *width* para o size classes *Regular* - **wR**. O objetivo era trabalhar com o iPhone 7 Plus e demais dispositivos que operam neste ambiente. Observe, na figura a seguir, que o Xcode identificou 12 variações com a configuração “Regular Width Devices” - wR:



Com esse ambiente selecionado, qualquer alteração que for efetuada nas Views do aplicativo irá se aplicar apenas a este ambiente. Podem ser novos elementos visuais, alterações de propriedades, *constraints*, etc.

Para sair do modo size classes, deve-se selecionar novamente o botão <Vary for Traits>.

Quando o aplicativo for executado, essas configurações específicas se aplicarão apenas para os dispositivos que se enquadrarem no ambiente **wR**.

21.5 EXERCÍCIO - Criando a nova experiência

Utilizando o recurso de *Size Classes*, iremos criar uma nova experiência para a *View* de **Temperatura**, aproveitando o espaço extra desta tela, no seguinte ambiente:

- iPhone 7 Plus
- Em modo Landscape

Para isso, vamos efetuar as seguintes configurações:

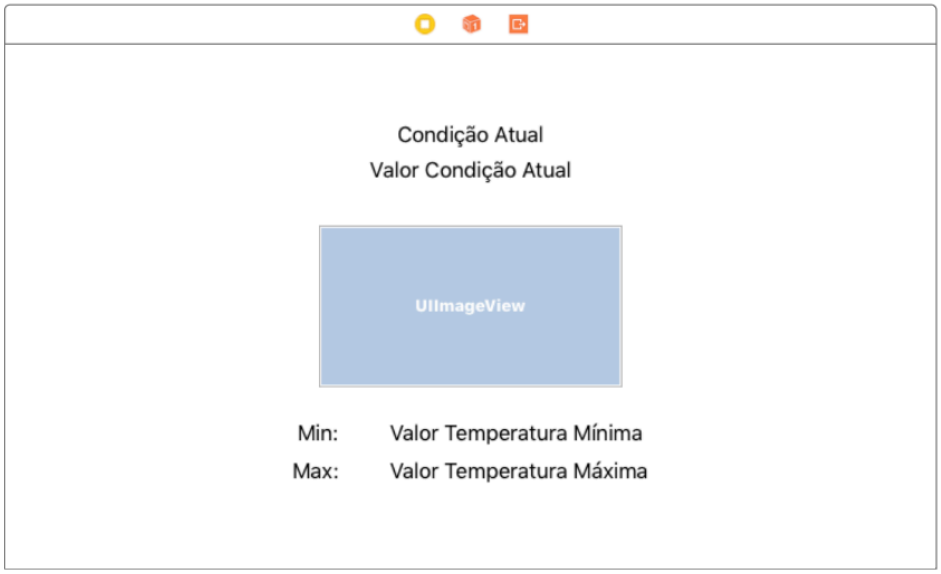
1. Abra o projeto regular do curso - **ContatosIP67.xcworkspace**

2. Selecione a **Main.storyboard**

3. Localize e selecione **Temperatura View Controller**

4. Utilizando a *Live View*, selecione o iPhone 7 Plus, e altere a orientação para *landscape*:

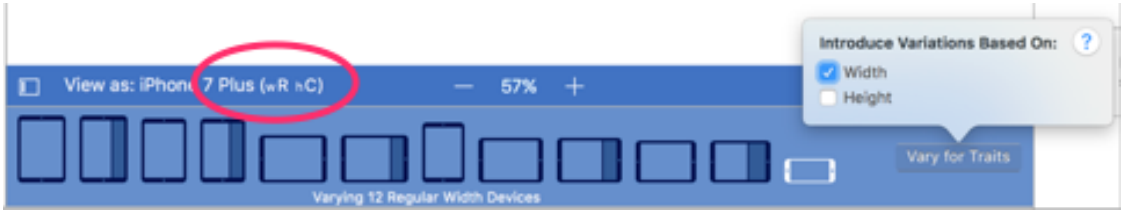
5. Após essa alteração, podemos observar que a nossa View realmente possui um espaço extra nas laterais, nesta orientação:



6. Como queremos criar a nova experiência apenas para o iPhone 7 Plus, em modo *landscape*, precisamos ativar o *size classes* neste ambiente.

7. Clique no botão <Vary For Traits>.

8. Como sabemos que a dimensão Regular é que denota espaço extra, iremos introduzir as variações apenas na dimensão de tamanho, ou seja wR, que é a dimensão do próprio iPhone Plus:

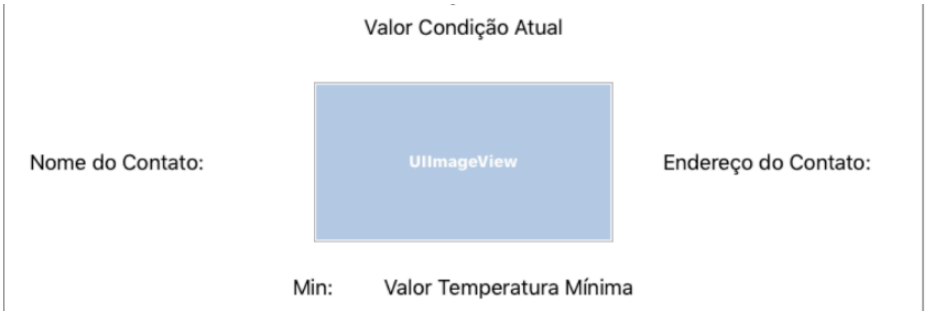


9. Com o *Size Classes* ativado, vamos agora construir a nova experiência de nosso app.

10. A nova experiência consiste em aproveitar esse espaço livre para inserir informações adicionais na tela. Iremos colocar o nome e o endereço do **contato**, de forma que o cliente possa visualizar essas informações diretamente na tela de temperatura, sem precisar retornar à tabela de contatos, caso queira consultar esses dados.

11. Utilizando a referência visual a seguir, crie 2 novos *Labels* nas extremidades da tela de temperatura.

12. Atribua aos *labels* as regras (*constraints*) de Auto Layout adequadas para manter o alinhamento e posicionamento em relação à *View* e demais elementos:



13. Os novos *labels* ficarão visíveis apenas para os devices que pertencerem a essa classe de devices, ou seja, ao ambiente **wR**.

14. Desative o Size Classes clicando novamente sobre o botão <Done Varying>.

15. Ok, sabemos que não está tudo pronto ainda, mas já podemos verificar se o *size classes* está funcionando.

16. Vamos então ver a mágica do Size Classes acontecer!

17. Utilizando a própria Live View, ou executando o App com outros simuladores, observe em que devices, e orientações, os novos *Labels* aparecem.

> Quando estiver utilizando o simulador, caso queira variar a orientação, utilize o atalho command + setas do telado

21.6 Verificando Size Classes no código!

Vimos a mágica do Size Classes acontecer, mas não estamos mostrando os dados reais do contato selecionado ainda.

Os *labels* devem ser conectados através de conexões IBOutlet, como fizemos em todo o nosso projeto. A diferença é que, no código, queremos condicionar a exibição dos dados apenas quando estivemos no ambiente adequado.

Para isso, utilizaremos mais um conceito bastante avançado de Auto Layout, que é a verificação do estado do Size Classes via código-fonte.

Utilizaremos a classe `UIUserInterfaceSizeClass` para descobrir, em tempo de execução em qual dimensão estamos em determinado momento - `horizontalSizeClass` (w) ou `verticalSizeClass` (h).

Lembrando que, em nossa experiência, precisaremos testar se estamos no ambiente do iPhone Plus, que é (wR hC).

21.7 EXERCÍCIO - Configurando e Personalizando a experiência

Neste exercício iremos expor os *labels* para atribuição dos dados de contato pelo código-fonte, porém faremos a atribuição dos dados apenas se estivermos no ambiente adequado.

Para o correto funcionamento do código, iremos utilizar recursos mais avançados e sofisticados da linguagem Swift, a saber:

- **Notificação** - iremos utilizar o sistema de notificações do iOS, para identificar as mudanças de orientação do dispositivo em tempo real
- **Extensão** - utilizaremos uma extensão da classe de temperatura para adicionar uma nova funcionalidade, que é a leitura do *size classes* da View Controller.
- **Tupla** - Como o retorno da função de size classes é duplo (uma das novidades da linguagem Swift), utilizaremos uma variável tipo **tupla** (outra novidade da Swift), para retornar os valores.

Inicialmente vamos configurar as conexões:

1. Selecione a **Temperatura View Controller**

2. Selecione o ambiente adequado de *size classes*, para que consiga visualizar os novos *labels* da tela.

3. Altere a classe **TemperaturaViewController.swift** para incluir duas conexões, permitindo que os *labels* possam ser atualizados com os dados do contato selecionado - nome e endereço.

4. Você precisará de 2 conexões, uma para exibir o nome e outra para exibir o endereço do contato selecionado:

```
// MARK: Outlets para modo Size Classes
@IBOutlet weak var labelNomeContato: UILabel!
@IBOutlet weak var labelEnderecoContato: UILabel!
```

Vamos agora personalizar a exibição das informações, atualizando os dados de contato na tela apenas se estivermos no ambiente adequado:

1. A função que irá retornar as dimensões do Size Classes será criada em uma extensão da própria classe **TemperaturaViewController**. Portanto, fora dos limites da classe, crie a seguinte extensão:

```
// Extensão para criar acessar ambiente de size class
extension UIViewController {

}
```

2. Dentro da extensão, iremos criar uma função que irá retornar as dimensões do Size Classes. Como são dois retornos (horizontal e vertical) iremos utilizar uma **tupla** para permitir o retorno duplo:

```
// Extensão para retornar ambiente de size classes
extension UIViewController {
    func sizeClass() -> (UIUserInterfaceSizeClass,
UIUserInterfaceSizeClass) {
```

```
        return(traitCollection.horizontalSizeClass,
traitCollection.verticalSizeClass)
    }
}
```

3. Agora, no âmbito da classe de temperatura, criaremos o *selector* que será responsável por atualizar os labels com os dados do contato:

```
    func orientationChanged() {
        // Verifica ambiente (size classes)
        // Carrega campos de contato apenas se estivermos num ambiente
wR (Plus e iPads em landscape)
        if sizeClass() == (UIUserInterfaceSizeClass.regular,
UIUserInterfaceSizeClass.compact) {
            labelNomeContato.text = contato?.nome
            labelEnderecoContato.text = contato?.endereco
        }
    }
```

4. Criaremos agora a notificação que irá ativar o selector recém criado, sempre que o usuário provocar uma mudança de orientação no dispositivo. Esse trecho de código pode ser inserido no início no evento `ViewDidLoad()`:

```
        // Cria notificação para detectar mudança de orientação em
tempo real
        NotificationCenter.default.addObserver(self, selector:
#selector(TemperaturaViewController.orientationChanged), name:
NSNotification.Name.UIDeviceOrientationDidChange, object: nil)
```

5. E, conforme as boas práticas de uso de notificações, iremos remover o *observer* da notificação no destrutor da classe. Em nosso caso, teremos que criar um destrutor:

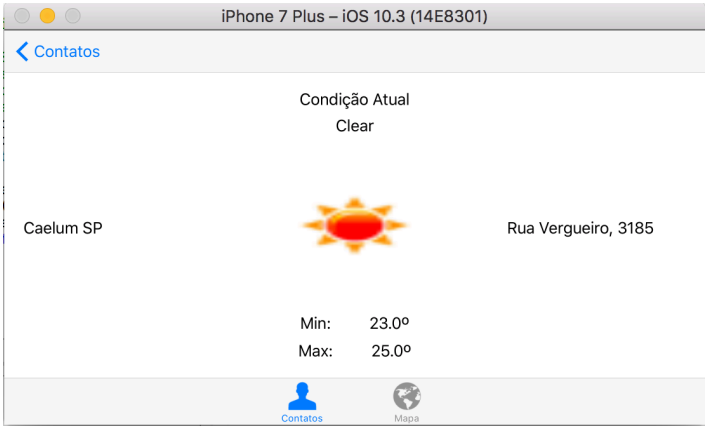
```
deinit {
    NotificationCenter.default.removeObserver(self)
}
```

6. Agora sim, devemos estar com tudo pronto!

7. Execute o aplicativo utilizando o simulador de algum iPhone Plus.

8. Quando a tela de temperatura for exibida, alterne a orientação do formulário, para verificar a mágica do *Size Classes*. Os novos *labels* devem aparecer em modo *landscape* e desaparecer em modo *portrait*.

9. O resultado final de sua versão do app deverá ser semelhante à figura a seguir:



DESAFIO
Em uma das turmas do curso, um aluno descobriu que, se entrarmos na tela de temperatura já em modo *landscape*, as informações dos *labels* de **nome do contato** e **endereço** não são atualizadas, permanecendo com seu conteúdo original.

- Você consegue identificar porque esse comportamento está acontecendo?
- Como resolver esse problema?